Building, Using, Sharing and Reusing Environment Concept Models

Christopher Chadbourne
VisiTech, Ltd.
3107 North 18th Street
Arlington, Virginia 22201
(703) 391-6264
chadbourne@visitech.com

Douglas Clark
Analysis and Technology, Inc.
2341 Jefferson Davis Highway
Arlington, Virginia 2202
(703) 418-8667
dclark@atinc.com

Keywords:
Simulation Implementation, Synthetic Environment, Unified Modeling Language

***ABSTRACT**: As Federations becoming larger and more complex, additional procedures and tools are being developed to help domain experts specify authoritative representations. This paper summarizes experience and lessons learned in developing a new tool needed by synthetic natural environment providers and simulation system integrators.*

The Environment Concept Model (ECM) is an object-oriented documentation technique. The technique is tailored for system engineers who must deliver a consistent synthetic environment representation, on time and within budget. The ECM documents the assumptions, features and limitations of environment data, effects and impacts, whether they are implemented as a single Federate or embedded within each Federate of a distributed Federation. The ECM leverages modern object-oriented design methodology, enables collaborative development of synthetic environment representations, and supports reuse using round trip software engineering principles.

Using the Unified Modeling Language as the reference modeling language, an example ECM is used to describe which features of object-oriented design languages are needed to develop an ECM. The paper further describes specific modeling tool features that support a full-featured ECM-building capability, to build ECMs that can be saved in repositories and reused to promote interoperability and reduce development time.

The paper describes how the ECM use case view records the objectives of the simulation, identifies the participants, and records other information which helps describe the context of the simulation application. It then describes the inferred environment view, the environment representation that is the sum of explicit and implicit environment-related requirements exposed by the use case. The paper then describes the implemented environment view, the representation that accounts for the level of "environment awareness" of simulation objects, the use of legacy and embedded environment data and software, and accommodations to the overall Federation development schedule and budget.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Building, using, Sharing and Reusing Environment Concept Models	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **VisiTech Ltd,3107 North 18th Street,Arlington,VA,22201**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
see report

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**10**	

1. Introduction

Based on the common need to create the basis for a unified environment representation, several major acquisition programs have cooperated to jointly develop a plan of action. The Maritime Virtual Environment Data Specification (MARVEDS) Initiative's objective is to develop the data specifications to enable a virtual prototype (VP) to interact with a maritime virtual environment. The goal is a verified, validated synthetic environment usable by multiple VPs throughout their lifecycle.

The ISD M&S Pilot Program is an Advanced Distributed Simulation Technology development program being conducted by the Program Executive Office for Theater Surface Combatants (PEO TSC). "The goal of the Integrated Ship Defense Modeling & Simulation Pilot Program is to develop and demonstrate a comprehensive modeling and simulation capability in support of the design and evaluation of components and systems directed towards phases of the ship defense mission, i.e., detect, control and engage. To this end, a comprehensive program plan was prepared in 1996 that details an approach to federating existing simulations in compliance with the High Level Architecture (HLA) to support PEO TSC system acquisition decisions." (Ref 1)

As Federations becoming larger and more complex, additional procedures and tools are being developed to help domain experts specify authoritative representations. In 1998 the authors, as members of a MARVEDS technical team, worked with the ISD M&S Pilot Program to develop a consistent synthetic environment representation for the ISD Phase 1 Federation (Ref 2). In the paper that documented our 1998 efforts we introduced the Environment Concept Model (ECM) document and generally described its intended features. This paper offers a more complete description of the ECM structure, and summarizes experience and lessons learned in developing the first instantiation of an ECM

2. Environment Concept Model Foundations

2.1 Why an Environment Concept Model?

Because environment representation, object models and object behaviors are so clearly linked, valid virtual prototype behaviors require consistent environment representations. Colloquially expressed, a consistent environment representation means that "everyone plays on the same day". More rigorously, consistent synthetic natural environments provide representations that are valid to a chosen resolution, and are spatially, temporally and spectrally continuous.

Responsive simulations must generate the prototype behaviors of interest, and they must be available in a timely fashion, for an investment that the program or project can afford. "Just enough" environment representation is matched to the needed behaviors, having been implemented within the constraints of the overall schedule and budget.

To achieve simulations where "everyone plays on the same day" with "just enough" environment representation, we make use of three important tools: the Environment Reference Framework, the Environment Reference Implementation Process, and the ECM.

Figure 1, the Environment Reference Framework, now under consideration by the Simulation Interoperability Standards Organization as a standard, provides a common basis for discussing environment representation in simulation. (The Environment Reference Framework is not a prescribed architecture, or a prescribed logical or physical simulation implementation.) Environment representations are created from combinations of environment data, effects/internal dynamics/impact calculations, and, possibly, heuristics. In whatever form, the representations must be combined with environment-aware military systems models. The resulting behaviors provide the simulation's value added output. Thus there is a direct, traceable relationship between the data and calculations that create the environment representation and the military systems behaviors.

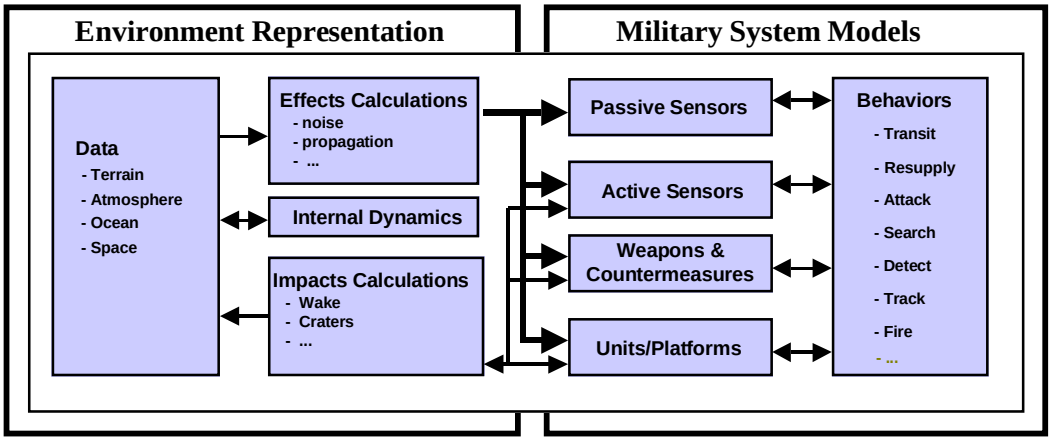

Figure 1. Environment Reference Framework

Figure 2, the Environment Reference Implementation Process, was developed specifically to support a program simulation system engineer in his effort to deliver responsive simulation capability on time and within budget. The process develops a recommended set of environment data, calculation models and heuristics that constitute "just enough" consistent environment representation to produce valid simulation results.

The process begins by reviewing the underlying operational scenario, and the participants (both real and modeled). If the simulation will be used in support of test range activities, then range instrumentation is reviewed as well. This initial review defines the application use case. Based on this review we can develop a unified environment representation (the inferred environment) that is the logical outgrowth of the objective-based use case. If necessary, we can then make recommendations for changes to scenarios or military systems models, if there is no other cost-effective way to satisfy environment representation requirements.

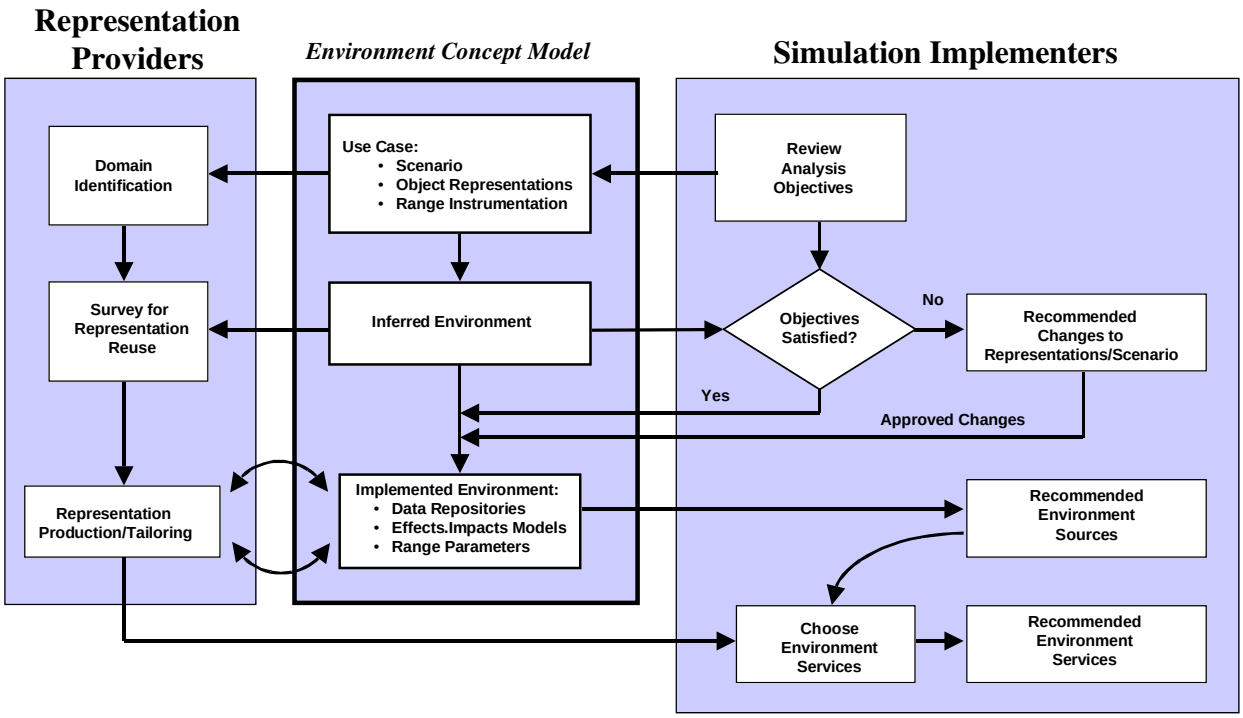

Figure 2. Environment Reference Implementation Process

The process continues by selecting the set of environment data, models and, if applicable, range measurement parameters that constitute the needed environment representation, specific to the simulation requirements at hand (the implemented view).

The environment selection process is accompanied by a documentation capability, the ECM, that complements sound engineering judgement with standards-based software modeling languages and tools. The ECM describes a unified environment representation (wherever it may occur) and makes it as accessible as every other component of the simulation implementation.

For us, the key point is that the ECM is the outgrowth of a collaborative process that coordinates the effort of simulation implementers and environment representation providers. Other reference frameworks might be appropriate. We have found, however, that any documentation approach should recognize the need for:

a. An incrementally constructed documentation approach that delivers partial products throughout the simulation implementation process

b. A use case description providing vital context information for the environment representation

c An environment representation description reflecting the use case needs, without compromise

d. A description of the environment representation as it is implemented in the simulation.

2.2 The Environment Concept Model Defined

The ECM is a unified description of the synthetic natural environment for a simulation application. The heart of the ECM is an object oriented analysis and design model, implemented in a standards-based object oriented analysis and design language. The object oriented model may be augmented by referenced electronic documents that amplify aspects of the representation. The object model may also be augmented by referenced files containing specific formatted queries and download requests from environment information repositories.

The core object oriented model describes computational processes and process interactions as well as static data structures. The purpose of the core object model is to unambiguously describe the environment representation to be used in the simulation application. The augmenting electronic documents and files may consist of reports, technical literature, briefings, test condition matrices, or other information that clarifies the reasoning behind selections of particular parameters or algorithms.

Because the ECM is not intended to be implementation-dependant, the ECM and its augments do not contain actual formatted environment data, source/executable code, or heuristics. However, in practical use, we have found that it is important to document allocations of environment representation (specific parameter files or calculation processes) to components of the simulation system. Further, by specifying a standards-based language for the object-oriented model, we anticipate greater future use of automated data schema and code generation, and code parsing to reverse engineer legacy code. Thus, the boundaries between ECM documentation and simulation content may blur as software engineering practice smoothes the evolution from design to runtime execution.

3. Developing and Using Environment Concept Models

3.1 Object Oriented Modeling Described

Object oriented modeling describes the physical world in an intuitive representation that can be directly replicated in software. Physical entities (objects) are described by their characteristics (attributes) and their behaviors (operations). The entities may be grouped into sets (classes) in heirarchies (subclasses and superclasses). The physical entities may exchange information (messages). Sometimes an object with its attributes and operations may represent an example of a category of physical entities (a stereotype or a type). Numbers of non-heirarchically related objects may be associated together (packages). When implemented in software, modules of code (components) may be installed (deployed) on one or more processing hosts.

Object oriented design languages use a descriptive notation that attempts to unambiguously define the emerging representation. Often, languages are both graphical and textual, with diagrams and terms having an agreed-upon grammar and syntax. Different types of symbols and notations are placed in different types of diagrams to describe (expose) the design from different perspectives (views).

Modern object oriented design tools to create sufficiently complete and unambiguous descriptions that the resulting files can be input to code generators

to create database designs or source code. Increasingly, design tools have parsers that are able to reverse engineer source code into design diagrams (round trip engineering).

We suggest two texts that will amplify this skeletal description. The first, Fowler's UML Distilled, (Ref 2) provides an overview of object oriented analysis and design from several viewpoints as well as introducing the reader to the Unified Modeling Language. The second, Booch/Rumbaugh/Jacobson's The Unified Modeling Language User Guide, (Ref 3) is a more focused, detailed explanation of the applications and conventions of the Unified Modeling Language (UML).

3.2 A Note about Object Oriented Modeling Tools

Building and using ECMs doesn't require using a particular object oriented language, or a particular documentation tool. However, whatever language is used, one should satisfy oneself that it can represent the application's environment representation in a way that communicates effectively with the entire simulation team. Similarly, one may choose from any number of documentation tools. For us, a good tool is tolerably easy to use, allows one to reuse all or part of ones previously developed ECMs, and integrates into the team's round trip engineering process. We use the Unified Modeling Language, developed by Rumbaugh, Booch and Jacobson and now being maintained by the Object Management Group.

To document our models, we have used Rational Corporation's Rational Rose. (Over a dozen vendors now offer UML modeling tools with varying features, in a range of prices and licensing conditions.) Our experience shows that a suitable tool should have, at minimum, the following features:

a. Follows the notation conventions established for the modeling language

b. Documents requirements and stakeholders explicitly, in addition to the environment representation schema

c. Documents dynamic (message passing, changing participants, etc.) as well as static (stakeholders, activities, physical objects to be simulated) aspects of the simulation scenario.

d. Provides a features for representing different views of the same simulation

e. Provides a capability to link external files with elements contained within the core object model file(s).

f. Provides graphical views and tools for creating and editing object model views

g. Provides file export and printing of selected object model views

In addition, a preferred modeling tool has the following additional features:

a. Exports and imports models views to and from a commonly accessible model repository

b. Generates data schemas, interface definitions, structured queries, and code structures from the object model

c. Parses database schemas or source code into object model content

d. Provides HTML file output of selected object model views

e. Provides automated modeling syntax checking, to avoid notation errors

Above all, good models create insight and understanding. That's the goal of the ECM, and the selected modeling tool should facilitate that goal.

3.3 Building and Using the Use Case

In UML, the use case view is used to show the intended behavior of the system to be implemented. Figure 3 shows the ISD Phase 1 Use Case view, as represented in Rational Corporation's Rational Rose modeling tool. We develop Use Case Diagrams to foster a common understanding between the federation system engineer, the federation developer, the environment system engineer, and the environment domain experts about the system being simulated. The Use Case Diagram explicitly describes the participants in the overall project (the actors), the processes to be simulated (the simulation activities), and other relevant project efforts that are not being simulated (the non-simulation activities)

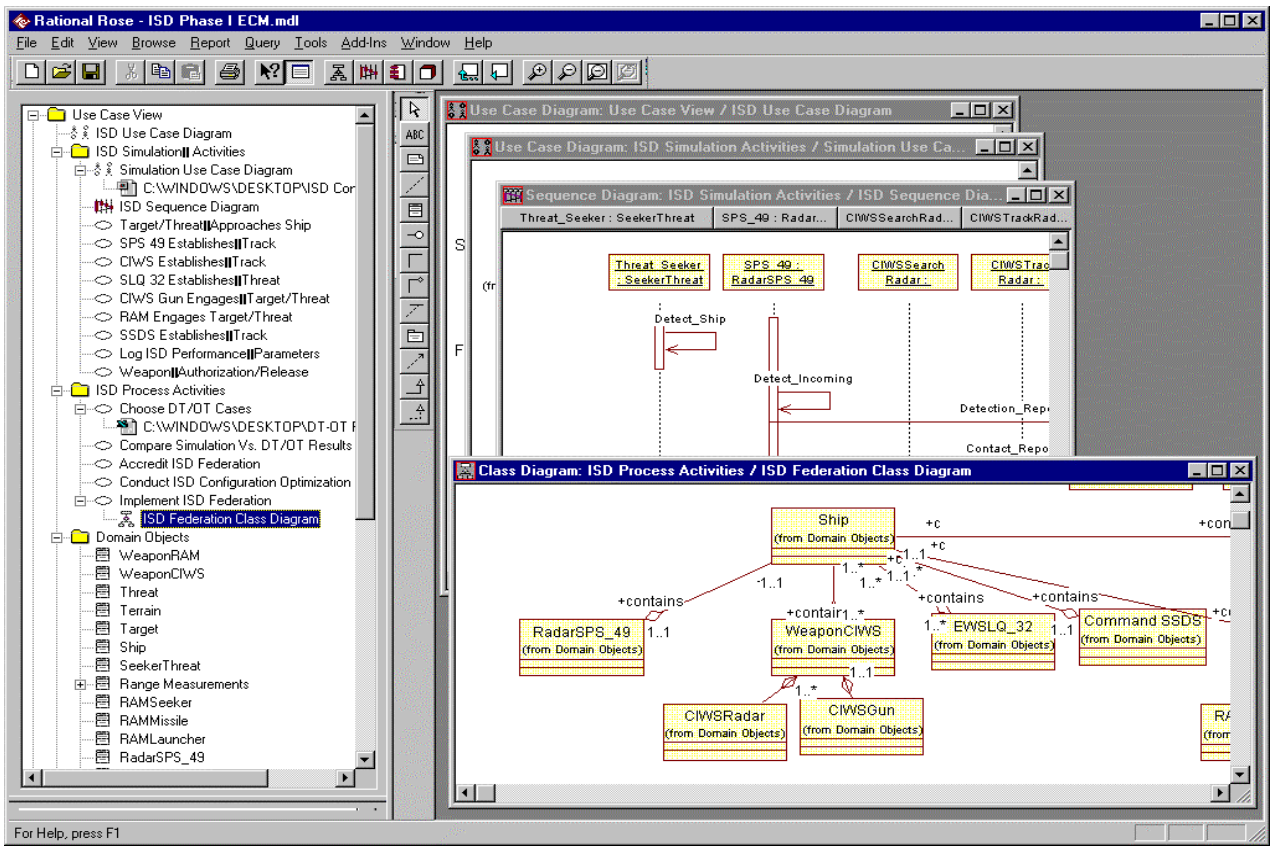

Figure 3. ISD Phase 1 Federation Use Case View

The Use Case Diagram records simulation activities because the activities answer the question "what activities are being modeled/simulated?" The simulation activities are groups of simulation events packaged as a short description. Every Use Case Diagram has a simulation activity entitled "Log simulation data" as a reminder that logging function must be explicitly considered in developing synthetic environment representations.

The simulation activity descriptions have associated lists of objects. These objects are the participants in the simulation: ships, aircraft, battalions, command structures, etc. Taken as a whole, the objects associated with the simulation activities correspond to the order of battle. The simulation activities and their associated objects are the basis for determining the size and scope of the battlespace. The Class Diagram is used to describe the participant objects comprising the scenario to be simulated. We capture limited information about the object as a way of determining which environment parameters affect object behaviors in the scenario. The Class Diagrams answer the question "What participant characteristics are affected by environment?"

The Sequence Diagram is used to represent the flow of activities over time to answer the question "How long is the simulation period?" The Sequence Diagram helps to scope environment representation requirements by identifying the information exchanged between objects, and an order for exchanging that information. Generally, we only construct diagrams using objects that are environment-sensitive, and we only identify environment-sensitive information exchanges.

However, the syntax of design languages (including UML), sometimes need a little help in communicating the essence of a tactical situation. Context Diagrams are simple schematics of a scenario, not created in a UML notation. The Context Diagram often looks like an annotated map or sensor display and is a shorthand means of communicating with operational domain experts accustomed to tactical or command system screen displays.

One may be tempted to use a Context Diagram as the only means of capturing the context of a simulation. If the scenario is simple; and the simulation is simple; and there is no intent to enhance the simulation; and no intent to reuse the simulation; then a Context Diagram

is all that's needed. But for the remaining 90% of simulation applications, we think Use Case, Class and Sequence Diagrams are a wise investment.

3.4 Building and Using the Inferred Environment View

There are actually two distinct environment representations under development in each project. The first is the environment representation inferred by the characteristics of the use case. By "inferred", we mean the representation that is a logical extension of the (a) level of fidelity appropriate to the purpose of the use case, (b) the domains and bandwidths in which the participating objects operate, and (c) the space and time described in the scenario. The second is the environment representation implemented for the simulation. The implemented environment describes the actual parameters, data sets and models implemented in the simulation.

The general approach to building an inferred view is to begin, if possible, with an existing environment representation structure, and then to edit and enhance that structure to satisfy the use case.

When building a first ECM, or for unique applications, one may wish to develop one's own stereotype class structure. For broadly based representations, we recommend a structure with superclasses for terrain, ocean, atmosphere as well as a superclasses for constants and a superclass for coordinate notations. Within each such superclass we would layer representation categories based on physical objects and processes, as appropriate. For instance a terrain superclass might have subclasses for topology, vegetation, cultural features, etc.

For more limited applications in a single domain such as Naval weapon system simulation, another approach may be easier to develop and use. In our continuing work, we have developed classes for various types of energy propagation phenomena such as scattering, specular reflection, dialectric values, and sea surface characterisations. We use this set of stereotypes because the environment-related processing in the ISD Phase 1 federation is almost exclusively associated with radio frequency and infrared energy propagation and reflection. (Ref 2)

Regardless of the scope of the class structure, our experience suggests that classes should always list attributes, but may or may not list operations. Operations (descriptors for algorithms or calculations) are often the determining factor in determining fidelity, and stereotypes are intended for a range of fidelities.

With the stereotype class structure in place, the next step is to use the class structure as a baseline, auditing the baseline against the needs as described in the use case. The audit proceeds by inspecting the class structure for consistency and completeness, changing the class structure as needed to achieve the environment representation that reflects the scenario and participants of the use case. The changes might include changes to the class hierarchy, new classes, class deletions, and changes to the class attributes. Most importantly, the class structure may be elaborated by adding operations (descriptions of calculations) to the class descriptions. The operations represent possible effects and impacts calculations, and they play an important role in determining the level of fidelity of the environment representation.

There are several common sources of inconsistency in environment representations, and these sources fall roughly into three categories. First, there may be inconsistencies between the representations for difference environment regimes, (terrain, atmosphere, oceans, the surf zone, etc.), with respect to length scales and boundary interfaces. Second, there may be inconsistencies between the time scale of simulation events and the time scale of the environment representation (static environment representations vice dynamic weather, etc.). Third, there may be inconsistencies in the representation of environment effects at different bandwidths throughout the energy spectrum.

The complete inferred view provides "just enough" environment representation to satisfy the needs of the simulation. This means that changes in environment state result in meaningful changes to the simulation object behaviors of interest. Naturally, the definition of "meaningful changes" is often entirely application-dependent. Thus the key to auditing the inferred environment view for completeness is to trace computational sequences back from the simulation behavioral outputs back through effects and impacts calculations, right through to environment data. There may be valid reasons why an object behavior is not at all sensitive to environment. Often, however, this insensitivity is due to gaps in the environment data or calculation capability; the absence of environment classes, or missing attributes operations within a class.

3.5 Building and Using the Implemented Environment View

We have found that there is often a difference between the environment representation inferred by the use case and the representation that is implemented for the simulation. The simulation systems engineer, responsible for balancing Federation implementation concerns, is often unable to provide all the resources needed to fully implement the inferred environment implementation. The system engineer's goal is to deliver a simulation capability that fulfills application needs; environment representation investments are balanced against hardware purchase needs, integration testing needs, etc. Differences can develop because of implementation schedule and cost constraints, lack of suitable data sets, inability to modify proprietary simulation software, to name just a few reasons.

The implemented environment representation view is the documentation of the actual environment representation to be used in the simulation at runtime. The implemented view is, most often, the final compromise between the consistent, complete inferred environment representation and the realities of implementation budgets, schedules, proprietary software rights, available data sets, certified software, etc.

The representation notation used for the implemented view is the same as for the inferred view: Class Diagrams and Interaction Diagrams. However, in the implemented view, we associate components of the environment data and code with components of the simulation, if the implementation is to be distributed.

The key differences in approach to developing the implemented view lie in whether the environment representation is being newly developed, or whether an existing representation is being modified for reuse. For newly developed environment representations the process moves forward from the inferred view, editing the implemented view to accommodate the overall simulation implementation process. For reused/modified representations, a data/model package is reverse engineered from data schemas and code into the object model syntax. Then the reverse engineered implemented view is compared with the inferred view, and the implemented view is edited as needed to approach the inferred view.

The Phase 1 ISD HLA-compliant federation contained environment data and calculations embedded in each federate's source code. As a result, as shown in Figure 4, we elected to group the implemented environment classes under packages representing the individual federates in the ISD Phase 1 federation.

Its often useful to develop the inferred and implemented views in parallel, and to compare them frequently. The emerging difference between the inferred and implemented views often provides valuable insight into sources of approximation in the simulation results. If the simulation system engineer is uncomfortable with the impact of the approximations, then there is still time to change the implemented environment to more fully reflect the inferred environment representation.

As with use case documentation, it's wise to offer an alternate description of the implemented view. For a use case alternative, we described the Context Diagram. As an alternative to the implemented view, we recommend the Consistent Environment Description. The Consistent Environment Description is a short textual document summarizing the implemented environment's impact on each simulation object, and any limitations on the object behavior validity caused by the scope of the implemented environment representation.

4. Sharing Environment Concept Models

The ECM is documented in a standards-based notation, using standards-based tools. As a result, representations documented in the ECM can be exported for several purposes. The ECM use case, inferred view and implemented view, can be exported to repositories to save as reusable representations, or to accompany reusable environment data and model sets. The ECM implemented view can also be used as input to object model template generators to quickly develop simulation object models or federation object models. Finally, the implemented view can be exported to application generators to generate database schemas or source code.

At this point the ECM content becomes implementation-specific, a part of the simulation software design and production process. Today, the particular export capabilities depend on which object oriented design and analysis tools is chosen to document the ECM. However, the software development trend is towards interoperable tools that can be chained to create an end-to-end software development environment.

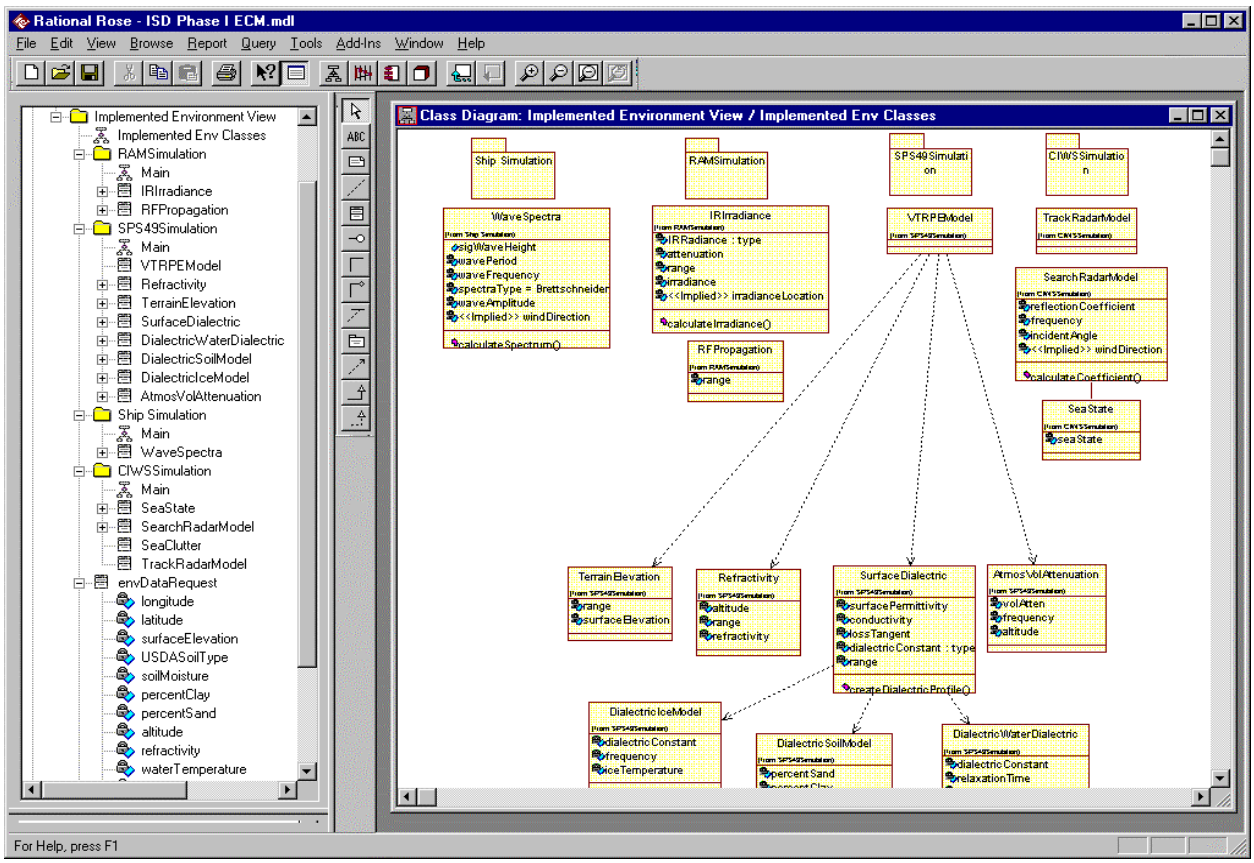

Figure 4. ISD Phase 1 Federation Implemented Environment View

If an organization or project already uses a suite of software engineering tools, it may be a worthwhile investment to choose a compatible ECM-building tool.

The ECM should be viewed as a living document, and maintained using the same mechanism as other project-related analysis, design and implementation documents. However, the nature of ECM content may complicate life cycle maintenance. First, the ECM uses both static and dynamic object notations. Many existing repositories were originally intended to manage static database designs, and store only the static portions of object oriented models. Second, an ECM may include referenced files, and may also include information in alternate formats (use case context diagrams and implemented view consistent environment descriptions). As a result there is no simple way of archiving a complete ECM, and no simple way of globally modifying all ECM content from a single entry point.

Our present approach to maintain the ECM as a separate entity, uploading portions of the ECM to archives and repositories for specific purposes.

5. Reusing the Environment Concept Model

Every use case is different, in some degree. Even when the objects and the scenario remain constant, the participants and analysis objectives will probably change. The use case view reflects these changes in the list of actors and non-simulation activities, shown in the use case diagram. It's important to consider the impact of different actors and non-simulation activities on the desired simulation output. As a result, we strongly advocate reusing legacy environment representation components… with care.

It's possible to assume the same use case and reuse an environment representation to achieve a consistent, authoritative environment. But there is a real risk that the reused environment representation is no longer responsive to the needs of the new analysis objectives. Therefore, don't reuse an environment representation without first (re)examining its associated use case. Changes to the use case will suggest changes in the inferred environment view, and the modification process adhering to the approach described previously.

However, reusing implemented environments, and editing the associated implemented environment view, may be more involved.

6. Acknowledgements

MARVEDS has been sponsored by the Navy Modeling and Simulation Management Office (N6M) and by the Naval Sea Systems Command . Project direction and management is by S.K. Numrich at the Naval Research Laboratory. The work cited in this paper was conducted with the support of the Program Executive Office for Theater Surface Combatants.

7. References

[1] PEO Theater Air Defense M&S Pilot Program Management Plan.

[2] Chadbourne, Christopher, Clark, Douglas, Neel, Timothy, "Insuring Consistent Synthetic Environmental Representation Across an Engineering Federation - A First User Case", 1998 Fall Simulation Interoperability Workshop

[2] Fowler, Martin, Scott, Kendall, UML Distilled, Addison-Wesley, 1997.

.

[3] Booch, Grady, Rumbaugh, James, Jacobson, Ivar, The Unified Modeling Language User Guide, Addison-Wesley, 1999.

Author Biographies

Christopher Chadbourne is President of VisiTech, Ltd. He supports the MARVEDS Working Group and is a technical team leader in the development of environment representation tools. For the past ten years Mr. Chadbourne has supported the Navy and joint simulation communities. Mr. Chadbourne has a BS degree in Engineering from the University of Michigan and an MBA from George Washington University.

Douglas Clark is a vice president of Analysis & Technology, Inc has over 25 years experience in modeling and simulation. He has lead efforts developing signature data and models to support simulations, developed design concepts and provided engineering support to the development of trainer systems and engineering federations. Currently he is a member of the MARVEDS Working Group concentrating on issues associated with the development and use of synthetic environments and also supports the Battle Force Tactical Trainer (BFTT) technical team. Mr. Clark has an MS in electrical engineering from the University of Connecticut.